Computer Vision

Day 3

2017, Tanta University
Eng. Sara Hussien

Previous Session

- Image Filtering:
  1. Box Filter [cv2.filter2D()]
  ```
  dst = cv2.filter2D(img,-1,kernel)
  ```
  2. Averaging
  ```
  blur = cv2.blur(img,(5,5))
  ```
  3. Gaussian Blurring: Gaussian blurring is highly effective in removing gaussian noise from the image.
  ```
  blur = cv2.GaussianBlur(img,(5,5),0)
  ```
  4. Median Blurring: this is highly effective against salt-and-pepper noise in the images
  ```
  median = cv2.medianBlur(img,5)
  ```

Edge Detection

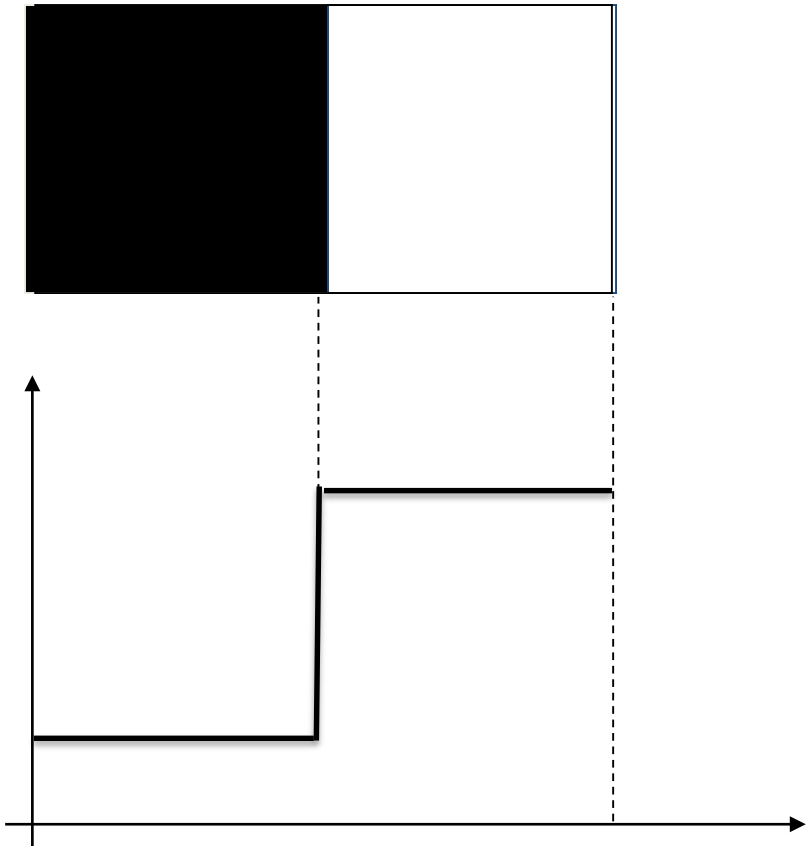
Edge Detection?

"The ability to measure gray-level transitions in a meaningful way."

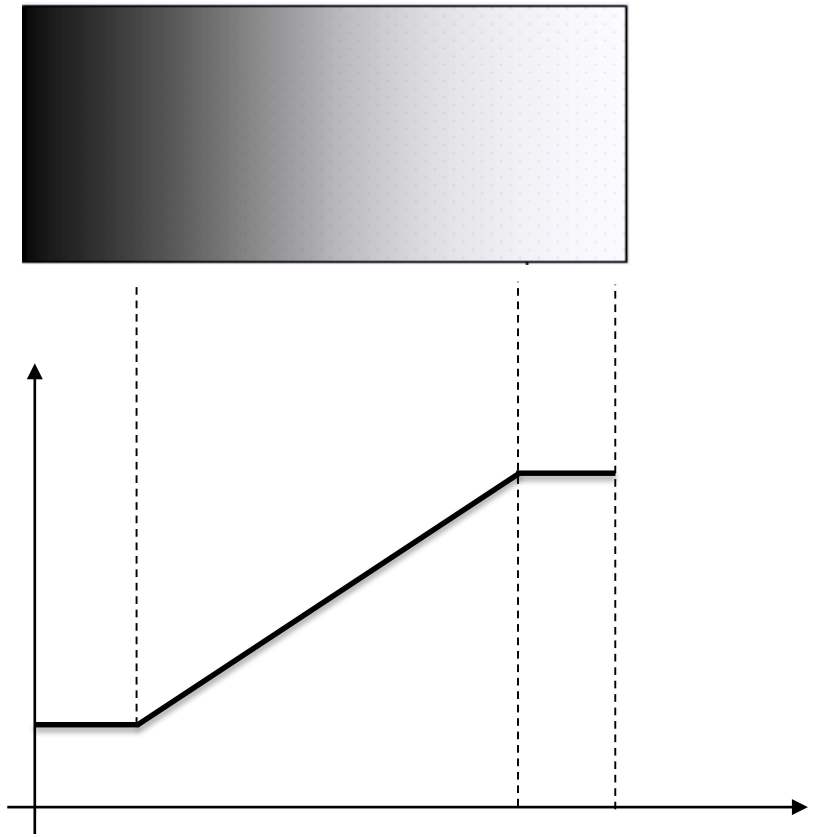
*(R.C. Gonzales & R. E. Woods – **Digital Image Processing**, 2nd Edition, Prentice-Hall, 2001)*

Gray-Level Transition

Ideal

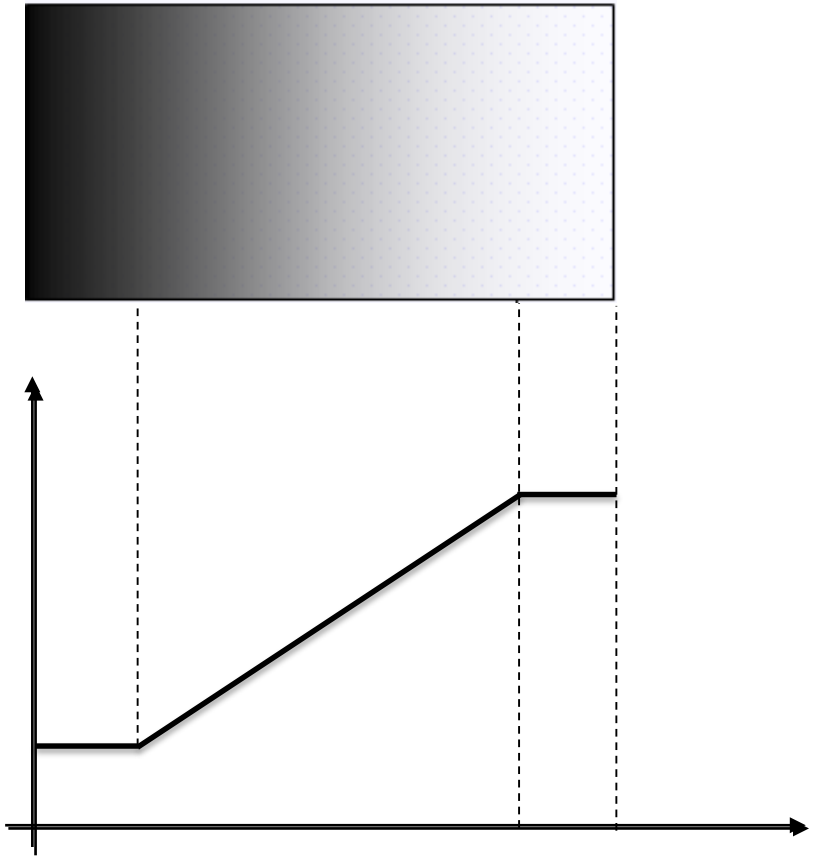


Ramp

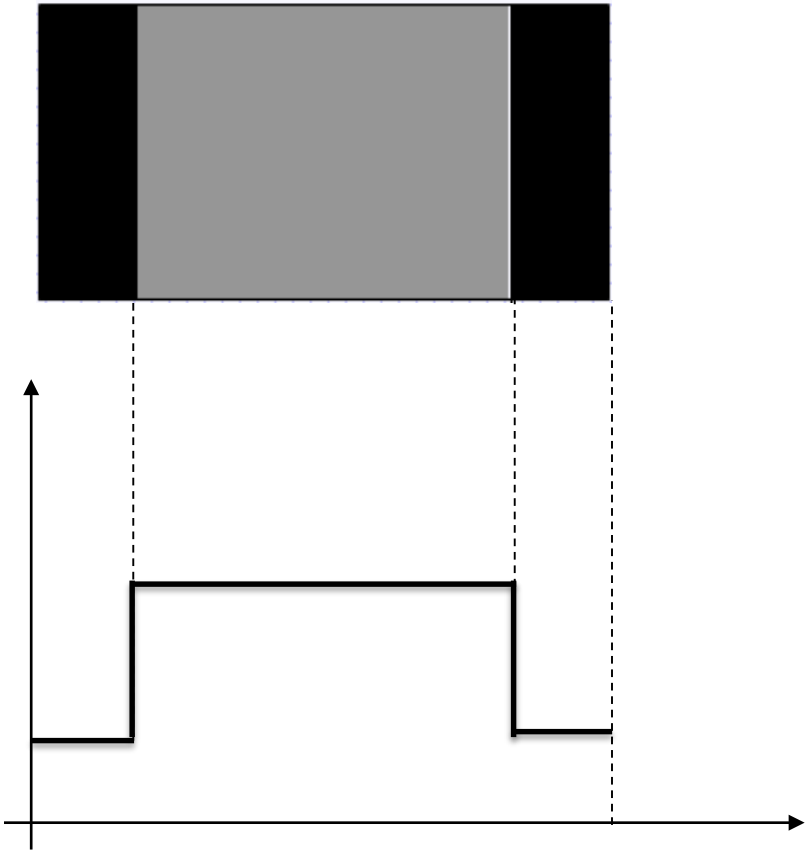


The First Derivative

Ideal

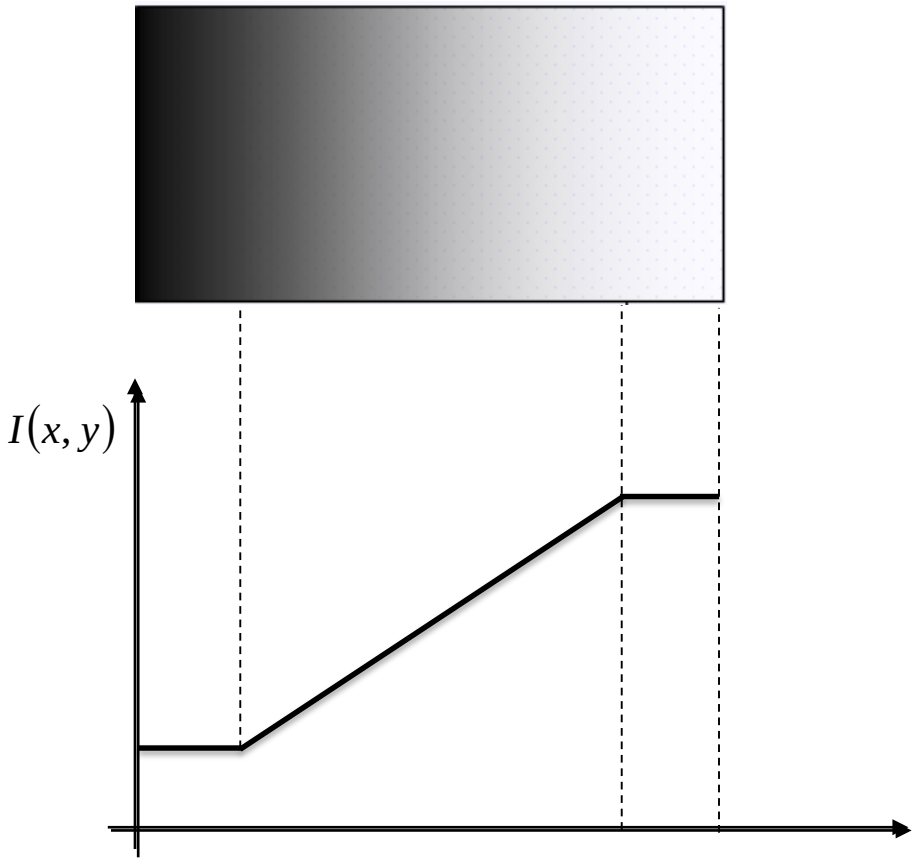


First Derivative

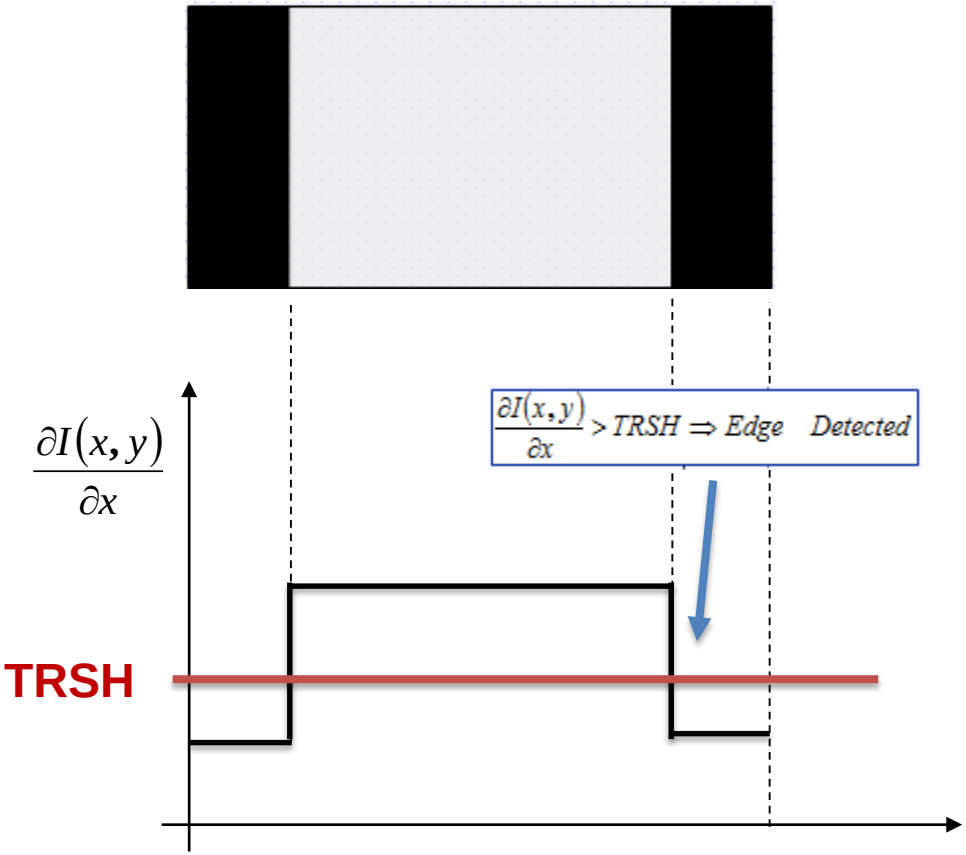


Detecting the Edge

Ideal

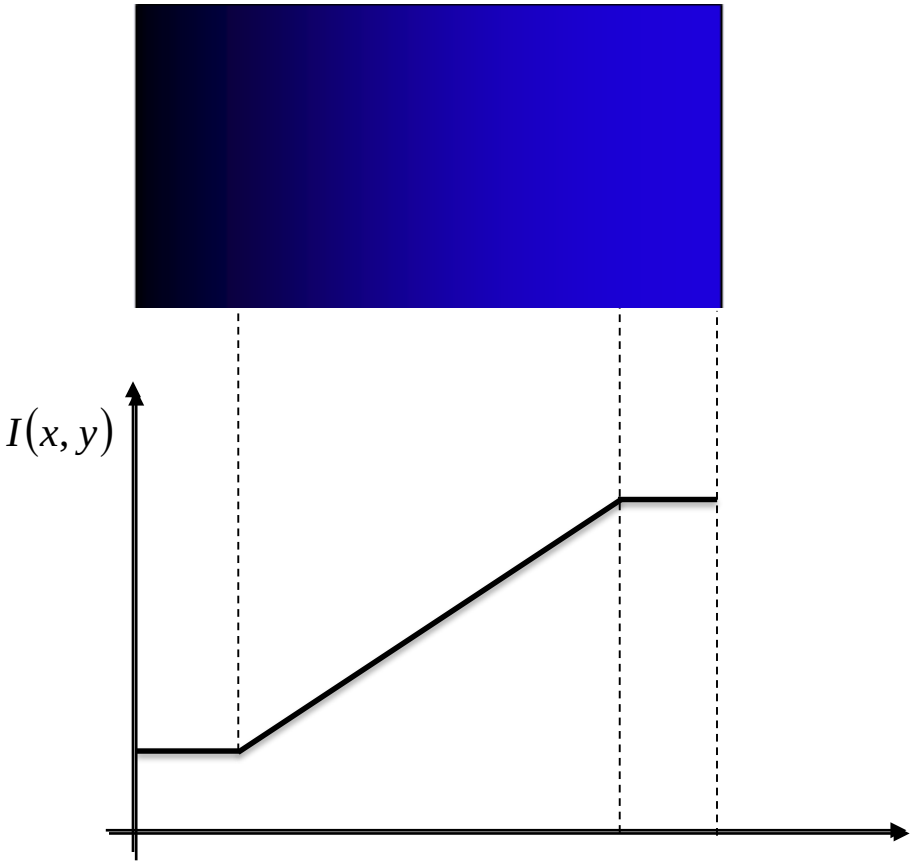


First Derivative

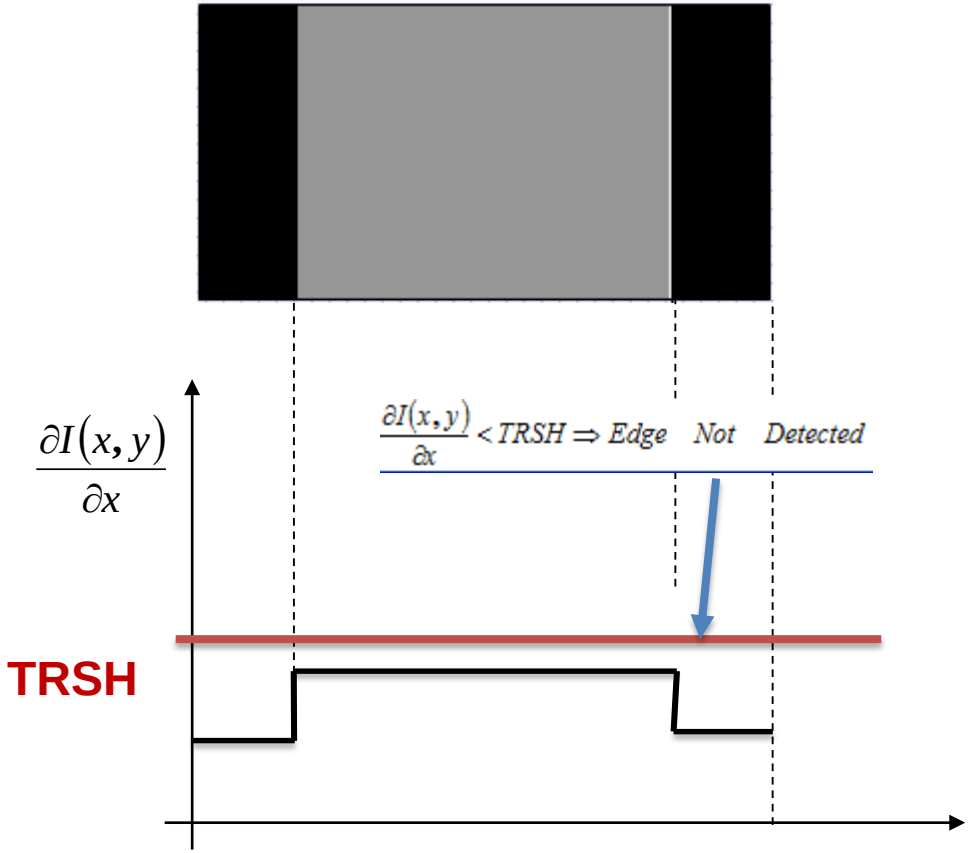


Detecting the Edge(2)

Ideal



First Derivative



Gradient Operators

- The gradient of the image *I(x,y)* at location *(x,y)*, is the vector:

$$\overline{\nabla I} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial I(x,y)}{\partial x} \\ \frac{\partial I(x,y)}{\partial y} \end{bmatrix}$$

- The magnitude of the gradient:

$$\nabla I = \left\| \overline{\nabla I} \right\| = \sqrt{\left[G_x^2 + G_y^2 \right]}$$

- The direction of the gradient vector:

$$\theta(x,y) = \tan^{-1}\begin{pmatrix} G_x \\ G_y \end{pmatrix}$$

The Meaning of the Gradient

- It represents the direction of the strongest variation in intensity

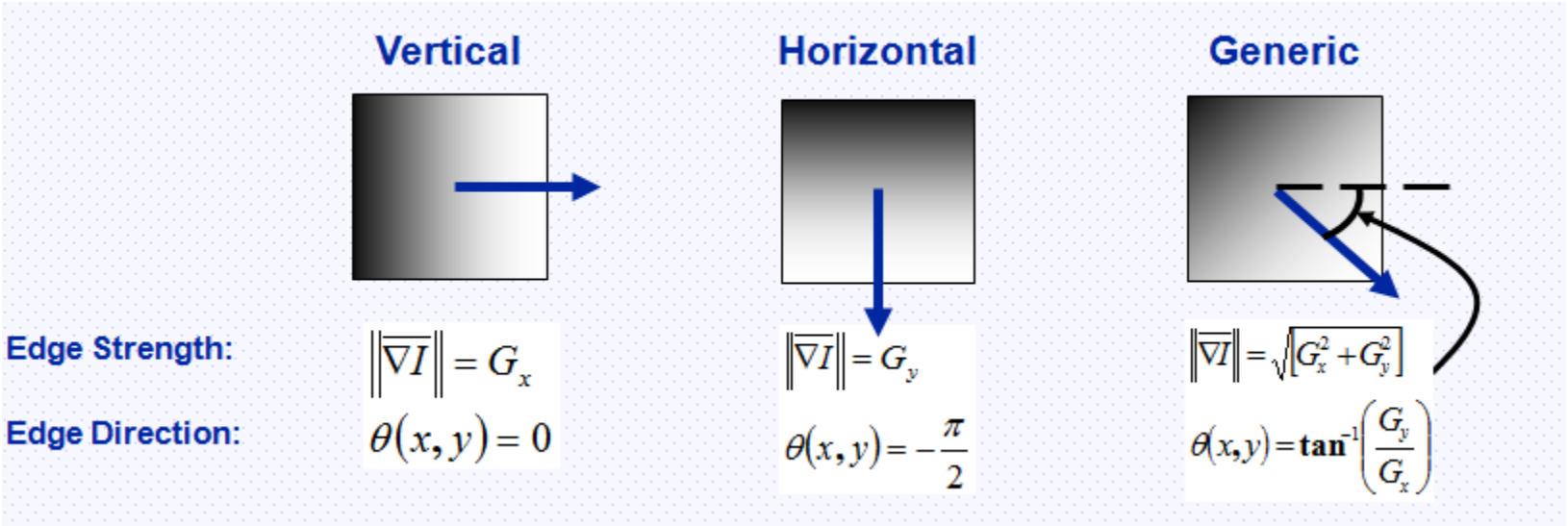


Image Gradients

- OpenCV provides three types of gradient filters or High-pass filters, **Sobel**, **Scharr** and **Laplacian**.

The Sobel Edge Detector

-1	-2	-1
0	0	0
1	2	1

Horizontal Edge

-1	0	1
-2	0	2
-1	0	1

Vertical Edge

Edge Detection Flow

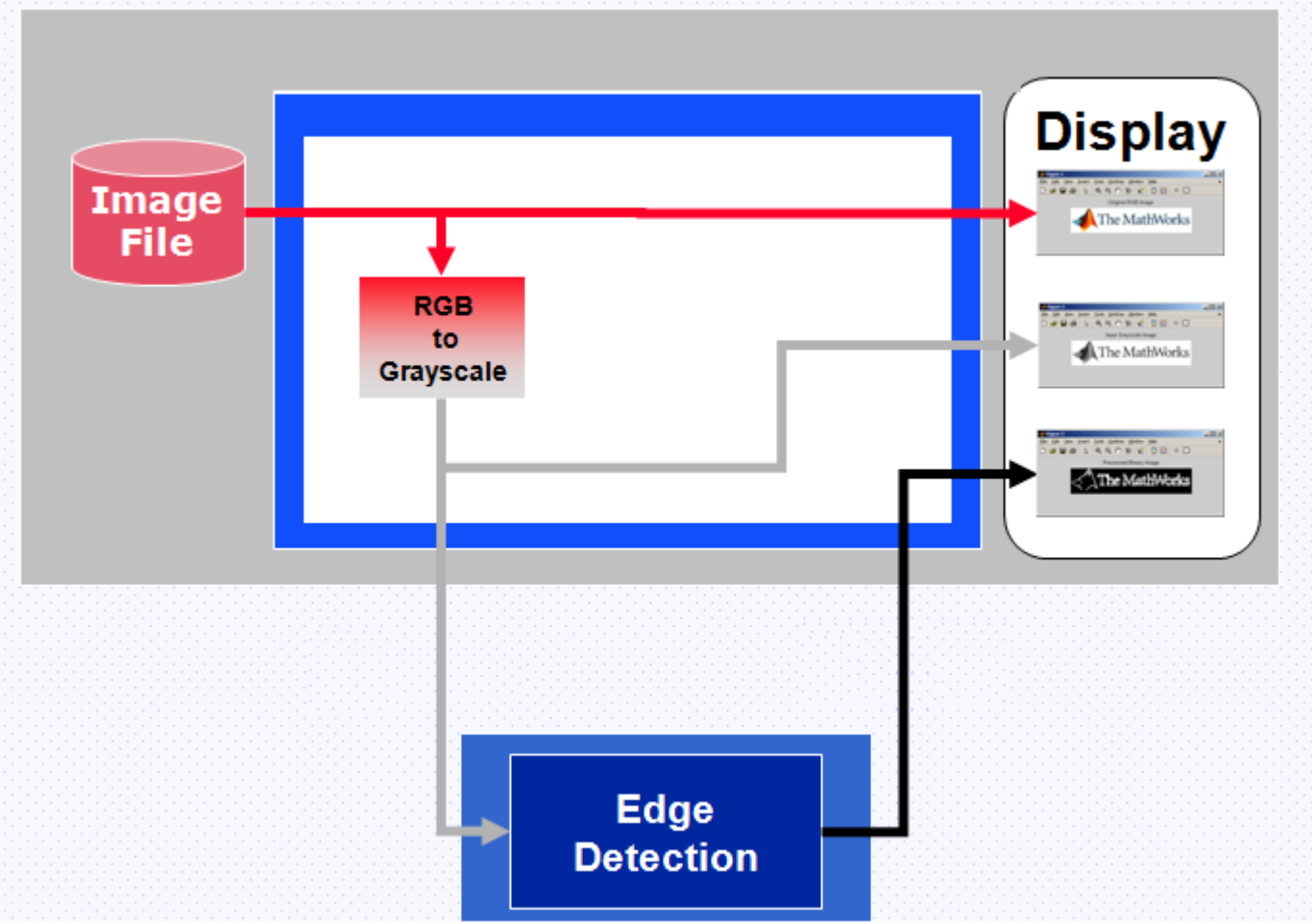


Image Gradients

- **Sobel Edge Detection**
  - void **Sobel**(InputArray src, OutputArray dst, int ddepth, int dx, int dy, int ksize=3, double scale=1, double delta=0, int borderType=BORDER_DEFAULT)

    Parameters:
    src – input image.
  - **dst** – output image of the same size and the same number of channels as src .
  - **ddepth** – output image depth; the following combinations of src.depth() and ddepth are supported:
    - src.depth() = CV_8U, ddepth = -1/CV_16S/CV_32F/CV_64F
    - src.depth() = CV_16U/CV_16S, ddepth = -1/CV_32F/CV_64F
    - src.depth() = CV_32F, ddepth = -1/CV_32F/CV_64F
    - src.depth() = CV_64F, ddepth = -1/CV_64F
  - when ddepth=-1, the destination image will have the same depth as the source; in the case of 8-bit input images it will result in truncated derivatives.

Sobel Edge Detection

- **xorder** – order of the derivative x.
- **yorder** – order of the derivative y.
- **ksize** – size of the extended Sobel kernel; it must be 1, 3, 5, or 7.
- **scale** – optional scale factor for the computed derivative values; by default, no scaling is applied (see getDerivKernels() for details).
- **delta** – optional delta value that is added to the results prior to storing them in dst.
- **borderType** – pixel extrapolation method (see borderInterpolate() for details).

```
# convolute with proper kernels
sobelx = cv2.Sobel(gray,cv2.CV_16S,1,0,ksize=3,scale=1,delta=0,borderType = cv2.BORDER_DEFAULT)
sobely = cv2.Sobel(gray,cv2.CV_16S,0,1,ksize=3,scale=1,delta=0,borderType = cv2.BORDER_DEFAULT)

abs_grad_x = cv2.convertScaleAbs(sobelx)    # converting back to uint8
abs_grad_y = cv2.convertScaleAbs(sobely)
```

The Canny Method

- The Canny method uses two thresholds, and enables the detection of two edge types: strong and weak edge.
- If a pixel's magnitude in the gradient image, exceeds the high threshold, then the pixel corresponds to a strong edge.
- Any pixel connected to a strong edge and having a magnitude greater than the low threshold corresponds to a weak edge.

```
detected_edges = cv2.GaussianBlur(gray,(3,3),0)
detected_edges = cv2.Canny(detected_edges,lowThreshold,lowThreshold*ratio,apertu
```

Code at Github: https://github.com/saraAwaad/ComputerVisionLab

References

- http://cs.brown.edu/courses/csci1430/
- https://www.udacity.com/course/introduction-to-computer-vision--ud810
- http://www.cc.gatech.edu/~afb/classes/CS4495-Fall2014/